Claims

[c1]   1. In a computer system, an improved method for developing and executing an application, the method comprising:

creating a model describing business objects and rules of the application;

creating source code for the application, including representing the model within the source code itself;

compiling the source code into an executable application;

running the executable application on a target computer in conjunction with a run-time framework that provides services to the executable application; and

while the executable application is running, reconstructing the model from the executable application and making it available to the run-time framework.

[c2]   2. The method of claim 1, wherein the model comprises a Unified Modeling Language (UML) model.

[c3]   3. The method of claim 1, wherein the source code is created using a programming language.

[c4]   4. The method of claim 3, wherein the programming lan-

guage is a selected one of Java, Pascal, and C#.

[c5]   5. The method of claim 3, wherein the programming language is one that supports reflection technique, thereby allowing reconstruction of the model at run-time from the executable application.

[c6]   6. The method of claim 1, wherein the reconstructed model is employed at run-time to support services that the run-time framework provides to the executable application.

[c7]   7. The method of claim 1, wherein the reconstructing step includes:
using reflection, reading metadata associated with the executable application to create a graph of code elements; and
spanning the graph for re-creating the model based on code elements encountered.

[c8]   8. The method of claim 7, wherein the spanning step includes:
as each code element is encountered, reconstructing a corresponding portion of the model.

[c9]   9. The method of claim 7, wherein the spanning step includes traversing the graph using a selected one of depth-first, breadth-first, and ad-hoc traversal tech-

nique.

[c10] 10. The method of claim 1, wherein the reconstructing step includes:

detecting a class having a package element; and

creating a corresponding Unified Modeling Language (UML) package for the reconstructed model.

[c11] 11. The method of claim 10, further comprising:

detecting an attribute specifying that a class belongs to the UML package; and

specifying in the reconstructed model that the class belongs to that UML package.

[c12] 12. The method of claim 1, further comprising:

after reconstructing the model at run-time, testing integrity of the reconstructed model.

[c13] 13. The method of claim 12, further comprising:

ensuring that all classes in the model belong to a common superclass.

[c14] 14. The method of claim 13, further comprising:

if all classes in the reconstructed model do not share a common superclass, automatically constructing a common superclass for those classes.

[c15] 15. The method of claim 1, wherein the reconstructed

model is stored in a cache memory available to the run-time framework.

[c16] 16. The method of claim 1, wherein the model is initially created using a modeling tool, and wherein the source code is compiled using a compiler.

[c17] 17. The method of claim 1, wherein the step of creating source code includes:

representing some information of the model in source code as language constructs.

[c18] 18. The method of claim 1, wherein the step of creating source code includes:

representing some information of the model in source code as attributes.

[c19] 19. The method of claim 18, wherein attributes comprise specifiers to structural code elements.

[c20] 20. The method of claim 1, wherein the step of creating source code includes:

representing some information of the model in code artifacts that exist expressly for carrying model information in source code.

[c21] 21. A computer-readable medium having processor-executable instructions for performing the method of

claim 1.

[c22]　22. A downloadable set of processor-executable instructions for performing the method of claim 1.

[c23]　23. In a computer system, an improved system for developing and executing an application, the system comprising:

a modeling tool for creating a model describing business objects and rules of the application;

a module for creating source code for the application and representing the model within the source code itself;

a compiler for compiling the source code into an executable application; and

a run-time framework that is able to reconstruct the model from the executable application and use it for providing services.

[c24]　24. The system of claim 23, wherein the model comprises a Unified Modeling Language (UML) model.

[c25]　25. The system of claim 23, wherein the source code is created using a programming language.

[c26]　26. The system of claim 25, wherein the programming language is a selected one of Java, Pascal, and C#.

[c27]　27. The system of claim 25, wherein the programming

language is one that supports reflection technique, thereby allowing reconstruction of the model at run-time from the executable application.

[c28] 28. The system of claim 23, wherein the reconstructed model is employed at run-time to support services that the run-time framework provides to the executable application.

[c29] 29. The system of claim 23, wherein the run-time framework includes submodules for reading metadata associated with the executable application to create a graph of code elements using reflection, and for spanning the graph for re-creating the model based on code elements encountered.

[c30] 30. The system of claim 29, wherein the submodule for spanning is able to reconstruct portions of the model based on corresponding code elements encountered in the executable application.

[c31] 31. The system of claim 29, wherein the submodule for spanning is able to traverse the graph using a selected one of depth-first, breadth-first, and ad-hoc traversal technique.

[c32] 32. The system of claim 23, wherein the run-time framework includes submodules for detecting a class

having a package element, and for creating a corre-
sponding Unified Modeling Language (UML) package for
the reconstructed model.

[c33]   33. The system of claim 32, further comprising:
a module for detecting an attribute specifying that a
class belongs to the UML package, and for specifying in
the reconstructed model that the class belongs to that
UML package.

[c34]   34. The system of claim 23, further comprising:
a submodule for testing integrity of the reconstructed
model.

[c35]   35. The system of claim 34, further comprising:
a submodule for ensuring that all classes in the model
belong to a common superclass.

[c36]   36. The system of claim 35, further comprising:
a submodule for automatically constructing a common
superclass for those classes when all classes in the re-
constructed model do not share a common superclass.

[c37]   37. The system of claim 23, wherein the reconstructed
model is stored in a cache memory available to the run-
time framework.

[c38]   38. The system of claim 23, wherein the model is initially

created using a UML modeling tool, and wherein the source code is compiled using a C# compiler.

[c39]   39. The system of claim 23, wherein the module for creating source code is able to represent some information of the model in source code as language constructs.

[c40]   40. The system of claim 23, wherein the module for creating source code is able to represent some information of the model in source code as attributes.

[c41]   41. The system of claim 40, wherein attributes comprise specifiers to structural code elements.

[c42]   42. The system of claim 23, wherein the module for creating source code is able to represent some information of the model in code artifacts that exist expressly for carrying model information in source code.

[c43]   43. A method for developing and executing an application on a computer system, the method comprising:
creating a model for developing an application using Unified Modeling Language (UML) technique;
generating source code to implement the model;
amending the source code for storing model information in the source code;
compiling the amended source code into an executable application and running the executable application on

the computer system;

reconstructing the model from the executable application; and

making the reconstructed model available for supporting operation of the executable application.

[c44] 44. The method of claim 43, wherein the source code is implemented using a programming language.

[c45] 45. The method of claim 44, wherein the programming language is a selected one of Java, Pascal, and C#.

[c46] 46. The method of claim 45, wherein the programming language is one that supports reflection technique, thereby allowing reconstruction of the model from the executable application.

[c47] 47. The method of claim 43, wherein the reconstructed model is employed by a run-time framework to provide services to the executable application.

[c48] 48. The method of claim 43, wherein the reconstructing step includes:

using reflection, reading metadata associated with the executable application to create a graph of code elements; and

spanning the graph for re-creating the model based on code elements encountered.

[c49] 49. The method of claim 48, wherein the spanning step includes:

as each code element is encountered, reconstructing a corresponding portion of the model.

[c50] 50. The method of claim 48, wherein the spanning step includes traversing the graph using a selected one of depth-first, breadth-first, and ad-hoc traversal technique.

[c51] 51. The method of claim 43, wherein the reconstructing step includes:

detecting a class having a package element; and

creating a corresponding Unified Modeling Language (UML) package for the reconstructed model.

[c52] 52. The method of claim 51, further comprising:

detecting an attribute specifying that a class belongs to the UML package; and

specifying in the reconstructed model that the class belongs to that UML package.

[c53] 53. The method of claim 43, further comprising:

after reconstructing the model, testing integrity of the reconstructed model.

[c54] 54. The method of claim 53, further comprising:

ensuring that all classes in the model belong to a common superclass.

[c55]   55. The method of claim 54, further comprising:
if all classes in the reconstructed model do not share a common superclass, automatically constructing a common superclass for those classes.

[c56]   56. The method of claim 43, wherein the reconstructed model is stored in a cache memory.

[c57]   57. The method of claim 43, wherein the model is initially created using a modeling tool, and wherein the amended source code is compiled using a compiler.

[c58]   58. The method of claim 43, wherein the step of amending the source code includes:
representing some information of the model in source code as language constructs.

[c59]   59. The method of claim 43, wherein the step of amending the source code includes:
representing some information of the model in source code as attributes.

[c60]   60. The method of claim 59, wherein attributes comprise specifiers to structural code elements.

[c61]   61. The method of claim 43, wherein the step of amend-

ing the source code includes:

representing some information of the model in code artifacts that exist expressly for carrying model information in source code.

[c62] 62. A computer-readable medium having processor-executable instructions for performing the method of claim 43.

[c63] 63. A downloadable set of processor-executable instructions for performing the method of claim 43.